REMARKS

By this Amendment, claims 1-3, 16, 23-25, 38, 45, 59, have been amended. No new matter has been added by these amendments.

Claims 1-65 are pending. Claims 1-65 are rejected. No claims have been canceled.

SUMMARY OF CLAIMED SUBJECT MATTER

Claims 1, 16, 23, 38, 45, and 59 are independent.

A. INDEPENDENT CLAIM 1

Claim 1 recites a method for managed object replication and delivery. [*See* ¶0023 *et seq.* & figs. 2 and 3(a)-3(c).] A request by a client for an object is directed to an edge server in a network. [¶0024; Fig. 2, 200; Fig. 3(a), step 300] If the edge server has the requested object, [¶¶0025-26; Fig. 3(a), step 305] then the requested object is served to the client, [¶¶0025-26; Fig. 2, 205; Fig. 3(a), step 310]. Otherwise, if the edge server does not have the requested object, the client request is redirected to a server that has the requested object [¶¶0028-29, Fig. 3(a), steps 320-345], and then the requested object is served to the client [¶0029, Fig. 2, 215, 230; Fig. 3(a), step 345]. If the requested object is popular, it is replicated to the edge server. [¶0029, Fig. 2, 220, 235; Fig. 3(a), step 330, Fig. 3(b), steps 350-360]

Claim 1	**Description in Specification**
A method for managed object replication and delivery, comprising:	
directing a request by a client for an object to an edge server in a network;	Fig. 1 shows the various components of embodiments of the invention in a high-level block diagram. ¶¶0003, 0020-22. Fig. 2 illustrates exemplary data flows according to embodiments of the invention. The initial request 200 from client 140 is directed to edge server 130. Fig. 2 & ¶0024 "Preferably, the client is directed to

Claim 1	**Description in Specification**
	an optimal edge server …".
if the edge server has the requested object,	Fig. 3(a), steps 305, 310.
serving the requested object to the client from the edge server;	Fig. 2, 205 shows the requested object being served from edge server 130 to client 140.
otherwise, if the edge server does not have the requested object, the edge server redirecting the client request to another server	Fig. 3(a), steps 320, 235, 345, 330
and serving the requested object to the client from the other server; and	Fig. 2, client request may be served from a parent server 120 (at 215) or from an origin server 110 (at 230). ¶0028-29
if the requested object is popular,	¶0029, Fig. 3(a), step 330, Fig. 3(b), steps 350-360. Popularity of an object may be measured, e.g., as described in the section titled "Determining Popularity," at ¶0035-40.
replicating the requested object to the edge server.	If popular. the requested object may be replicated to the edge server 130 from a parent server 120 or an origin server 110 (*See* figs. 1-2). Fig. 2 shows, at 235, replication to the edge server from the origin server ("the edge server initiates the replication (at 235, 360) of the popular requested object to the edge server from the origin server having the requested object" ¶0033). *See also* Fig. 3(b), 360. Fig. 2 also shows, at 220, replication to the edge server 130 from a parent server 120. ("the edge server initiates the replicating (at 220, 360) of the popular requested object to the edge server from a parent server that has the requested object." ¶0032.)

B. INDEPENDENT CLAIM 16

Claim 16 describes a method for managed object replication and delivery. A request by a client for an object is directed to an optimal edge server in a network. If the edge server has the requested object, it is served to the client. Otherwise (if the edge server does not have the requested object), the client request is redirected to a parent server in the network, and it is served to the client from that parent server. If the requested object is popular, it is replicated to the edge server from the parent server.

Claim 16	**Description in Specification**
16. A method for managed object replication and delivery, comprising:	
directing a request by a client for an object to an optimal edge server in a network;	Fig. 1 shows the various components of embodiments of the invention in a high-level block diagram. ¶¶0003, 0020-22. Fig. 2 illustrates exemplary data flows according to embodiments of the invention. The initial request 200 from client 140 is directed to edge server 130. Fig. 2 & ¶0024 "Preferably, the client is directed to an optimal edge server ...".
if the edge server has the requested object,	Fig. 3(a), steps 305, 310.
serving the requested object to the client from the edge server;	Fig. 2, 205 shows the requested object being served from edge server 130 to client 140.
otherwise, if the edge server does not have the requested object, the edge server redirecting the client request to a parent server in the network, and	Fig. 3(a), steps 320, 235, 345, 330
serving the requested object to the client from the parent server; and	Fig. 2, 215 ¶0028-29 "The parent server that has the requested object serves (at 215, 345) the object to the client"
if the requested object is popular,	Fig. 3(b), step 360. Popularity of an object may be measured, e.g., as described in the section titled "Determining Popularity," at ¶0035-40.
replicating the requested object to the edge server from a parent server in the network.	Fig. 2, 220, "the edge server initiates the replicating (at 220, 360) of the popular requested object to the edge server from a parent server that has the requested object." ¶0032.

C. INDEPENDENT CLAIM 23

Claim 23 recites a computer program product including computer program code to cause a processor to perform a method for managed object replication and delivery. The method performed by the program includes the same steps as the method of independent claim 1. A request by a client for an object is directed to an edge server in a network. If the edge server has the requested object, it is served to the client. Otherwise, the client request is redirected to a server that has the requested object and it is served to the client. If the requested object is popular, it is replicated to the edge server.

Claim 23	**Description in Specification**
A computer program product including computer program code to cause a processor to perform a method for managed object replication and delivery, the method comprising:	¶¶0062-66.
directing a request by a client for an object to an edge server in a network;	Fig. 1 shows the various components of embodiments of the invention in a high-level block diagram. ¶¶0003, 0020-22. Fig. 2 illustrates exemplary data flows according to embodiments of the invention. The initial request 200 from client 140 is directed to edge server 130. Fig. 2 & ¶0024 "Preferably, the client is directed to an optimal edge server …".
if the edge server has the requested object, serving the requested object to the client;	Fig. 3(a), steps 305, 310. Fig. 2, 205 shows the requested object being served from edge server 130 to client 140.
otherwise, the edge server does not have the requested object, the edge server redirecting the client request to another server, and serving the requested object to the client from the other server;	Fig. 3(a), steps 320, 235, 345, 330 Fig. 2, client request may be served from a parent server 120 (at 215) or from an origin server 110 (at 230). ¶0028-29
if the requested object is popular,	¶0029, Fig. 3(a), step 330, Fig. 3(b), steps 350-360. Popularity of an object may be measured, e.g., as described in the section titled "Determining Popularity," at ¶0035-40.
replicating the requested object to the edge server.	If popular. the requested object may be replicated to the edge server 130 from a

Claim 23	**Description in Specification**
	parent server 120 or an origin server 110 (*See* figs. 1-2). Fig. 2 shows, at 235, replication to the edge server from the origin server ("the edge server initiates the replication (at 235, 360) of the popular requested object to the edge server from the origin server having the requested object" ¶0033). *See also* Fig. 3(b), 360. Fig. 2 also shows, at 220, replication to the edge server 130 from a parent server 120. ("the edge server initiates the replicating (at 220, 360) of the popular requested object to the edge server from a parent server that has the requested object." ¶0032.)

D. INDEPENDENT CLAIM 38

Independent claim 38 recites a computer program product including computer program code to cause a processor to perform a method for managed object replication and delivery. The method performed by the program includes the same steps as the method of independent claim 16, *viz*, a request by a client for an object is directed to an optimal edge server in a network. If the edge server has the requested object, it is served to the client. Otherwise (if the edge server does not have the requested object), the client request is redirected to a parent server in the network that has the requested object, and it is served to the client from that parent server. If the requested object is popular, it is replicated to the edge server from the parent server.

Claim 38	**Description in Specification**
A computer program product including computer program code to cause a processor to perform a method for managed object replication and delivery, the method comprising:	¶¶0062-66.
directing a request by a client for an object to an optimal edge server in a network;	Fig. 1 shows the various components of embodiments of the invention in a high-level block diagram. ¶¶0003, 0020-22. Fig. 2 illustrates exemplary data flows according to embodiments of the

Claim 38	**Description in Specification**
	invention. The initial request 200 from client 140 is directed to edge server 130. Fig. 2 & ¶0024 "Preferably, the client is directed to an optimal edge server …".
if the edge server has the requested object, then	Fig. 3(a), steps 305, 310.
serving the requested object to the client from the edge server;	Fig. 2, 205 shows the requested object being served from edge server 130 to client 140.
otherwise, if the edge server does not have the requested object redirecting the client request to a parent server in the network and	Fig. 3(a), steps 320, 235, 345, 330
serving the requested object to the client from the parent server; and,	Fig. 2, 215 ¶0028-29 "The parent server that has the requested object serves (at 215, 345) the object to the client"
if the requested object is popular,	Fig. 3(b), step 360. Popularity of an object may be measured, e.g., as described in the section titled "Determining Popularity," at ¶0035-40.
replicating the requested object to the edge server from a parent server in the network.	Fig. 2, 220, "the edge server initiates the replicating (at 220, 360) of the popular requested object to the edge server from a parent server that has the requested object." ¶0032.

E. INDEPENDENT CLAIM 45

Independent claim 45 recites a system for managed object replication and delivery. The system includes a plurality of edge servers (Fig. 1, edge servers 130) in a network 100; and a plurality of parent servers (12) in the network (100). At least one of the plurality of edge servers (130) and the plurality of parent servers (120):

- direct a request by a client for an object to an edge server (130 in Fig. 2) in the network (also flow 200 in Fig. 2),

- if the edge server has the requested object (step 305, Fig. 3(a)), serve the requested object to the client from the edge server (205 in Fig. 2; step 310 in Fig. 3(a)),
- otherwise, redirect the client request to another server (120, 110) (step 320 in Fig. 3(a)) and serve the requested object to the client (215, 230 in Fig. 2; step 345 in Fig. 3(a)),
- if the requested object is popular (steps 315, 330 in Fig. 3(a), step 330 in Fig. 3(b)), replicate (220, 235 in Fig. 2; step 360 in Fig. 3(b)) the requested object to the edge server (130).

Claim 45	**Description in Specification**
A system for managed object replication and delivery, comprising:	
a plurality of edge servers in a network; and	Fig. 1, edge servers 130
a plurality of parent servers in the network,	Fig. 1, parent servers 120
wherein at least one of the plurality of edge servers and the plurality of parent servers:	Figs. 2, 3(a), 3(b)
direct a request by a client for an object to an edge server in the network,	130 in Fig. 2; 200 in Fig. 2
if the edge server has the requested object,	Step 305; Fig. 3(a);
serve the requested object to the client from the edge server,	205 in Fig. 2; step 310 in Fig. 3(a)
otherwise, if the edge server does not have the requested object, redirect the client request to another server, and	Step 320 in Fig. 3(a)
serve the requested object to the client, and	215, 230 in Fig. 2; step 345 in Fig. 3(a)
if the requested object is popular,	Steps 315, 330 in Fig. 3(a), step 330 in Fig. 3(b). Popularity of an object may be measured, e.g., as described in the section titled "Determining Popularity," at ¶0035-40.
replicate the requested object to the edge server.	220, 235 in Fig. 2; step 360 in Fig. 3(b)

F. INDEPENDENT CLAIM 59

Independent claim 59 recites a system for managed object replication and delivery. The system includes a plurality of edge servers (Fig. 1, edge servers 130) in a network 100; and a plurality of parent servers (12) in the network (100). At least one of the plurality of edge servers (130) and the plurality of parent servers (120):

- direct a request by a client for an object to an optimal edge server (130 in Fig. 2) in the network (also flow 200 in Fig. 2);
- if the optimal edge server has the requested object (step 305, Fig. 3(a)), serve the requested object to the client from the optimal edge server (205 in Fig. 2; step 310 in Fig. 3(a));
- otherwise, redirect the client request to a parent server (120) in the network (100) and serve the requested object (215 in Fig. 2; step 345 in Fig. 3(a)) to the client (140) from the parent server (120);
- if the requested object is popular (steps 315, 330 in Fig. 3(a), step 330 in Fig. 3(b)), replicate (220) the requested object to the edge server (130) from the parent server (120).

Claim 59	**Description in Specification**
A system for managed object replication and delivery, comprising:	
a plurality of edge servers in a network; and	Fig. 1, edge servers 130
a plurality of parent servers in the network,	Fig. 1, parent servers 120
wherein at least one of the plurality of edge servers and the plurality of parent servers:	Figs. 2, 3(a), 3(b)
direct a request by a client for an object to an optimal edge server in the network;	130 in Fig. 2; 200 in Fig. 2; Step 300 in Fig. 3(a); ¶0024 ("Preferably, the client is directed to an optimal edge server …")
if the optimal edge server has the requested object, then	Step 305; Fig. 3(a);
serve the requested object to the client;	205 in Fig. 2; step 310 in Fig. 3(a)

Claim 59	**Description in Specification**
otherwise, if the optimal edge server does not have the requested object, then redirect the client request to a parent server in the network; and	Step 320 in Fig. 3(a); 120 in Fig. 2.
serve the requested object to the client from the parent server;	215 in Fig. 2; step 345 in Fig. 3(a)
if the requested object is popular,	Steps 315, 330 in Fig. 3(a), step 330 in Fig. 3(b). *See also*, "Determining Popularity," at ¶0035-40.
replicate the requested object to the edge server from the parent server.	220 in Fig. 2; step 360 in Fig. 3(b)

THE PRIOR ART REJECTIONS

Claims 1-65 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Jungck (US. Pub. No. 2005/0021863 – hereinafter "Jungck") and Sim (U.S. Pub. No. 2003/0031176 – hereinafter "Sim").

ARGUMENT

A. BACKGROUND OF THE TECHNOLOGY

This invention relates to managed object replication and delivery and to content delivery networks (CDNs). As noted in the application at ¶0010, a

> typical content delivery network (CDN) operator deploys one or more parent servers, hosting a plurality of objects, in a network and one or more edge servers at the edge of the network to facilitate more cost-effective and efficient delivery of such objects to an end-user (client). End-users or client proxies that access customers' objects are called clients. Content provider companies, organizations, etc. that subscribe to the CDN service are referred to as customers. * * * It is typically desirable to serve objects from edge servers because the edge servers are typically closer (by various measures of distance) to end-users.

It is generally not desirable (or even feasible) to pre-populate edge servers with all of the content that may be requested of them. *Specification,* ¶0011.

> The main difficulty is due to the fact that many such

> objects are very large .... The ... space required to accommodate often large and sometimes rarely requested objects at every edge server can be cost prohibitive as the number of customers grows and the number of their objects increases. It may not even be possible to store a good working set of objects, for example a set of objects thought to be requested often and/or better suited to be served from an edge server, because of the size and changing demand for objects in the working set.

Id. It is also generally not desirable to try to *pre-populate* edge servers with objects for which there will likely be a high demand. *Id.* at ¶0012. As the inventors note in the Application, "it is difficult to predict popularity and difficult to manage pre-populating." *Id.*

Some prior solutions fetch objects *on demand. Id.* at ¶0013. "In such schemes, when a requested object is not available on a handling edge server, a connection is made between a parent server having the requested object and the handling edge server to fetch the requested object from the parent server. Such fetching suffers however from having to go through the parent path (the network path between the handling edge server and the parent server with the object) whenever a client requests an object that is not already at the particular edge server." *Id.* There are other problems associated with on-demand fetching through a parent server. As noted in the Application, at ¶¶0014-15, "[f]etching a large object to the handling edge server through a parent path can be slow." In addition, with *on-demand* population of edge servers, the client would have to wait for the object to reach the edge server before the object is served to the client.

Understanding these problems, the inventors were the first to realize the value of populating CDN caches with *popular* content using a dynamic measure of popularity "without having to make the end-user wait for such population." *Id.* at ¶0016.

Accordingly, in some aspects, this invention provides for methods and systems that intelligently replicate objects to edge servers if the objects are popular enough. Likewise, an object may be is removed from an edge server when the object is no longer popular. Furthermore, with the presently claimed invention, requesting clients do not have to wait for content to be moved to edge servers before being served (a client request will be handled by a server that has the requested content).

A "typical content delivery network (CDN) operator deploys one or more parent servers, hosting a plurality of objects, in a network and one or more edge servers at the edge of the network to facilitate more cost-effective and efficient delivery of such objects to an end-user (client)." *Specification* ¶0010. A block diagram of an exemplary topology of the managed object replication and delivery method and system according to embodiments of the invention is shown in Figure 1 of the application (reproduced below).

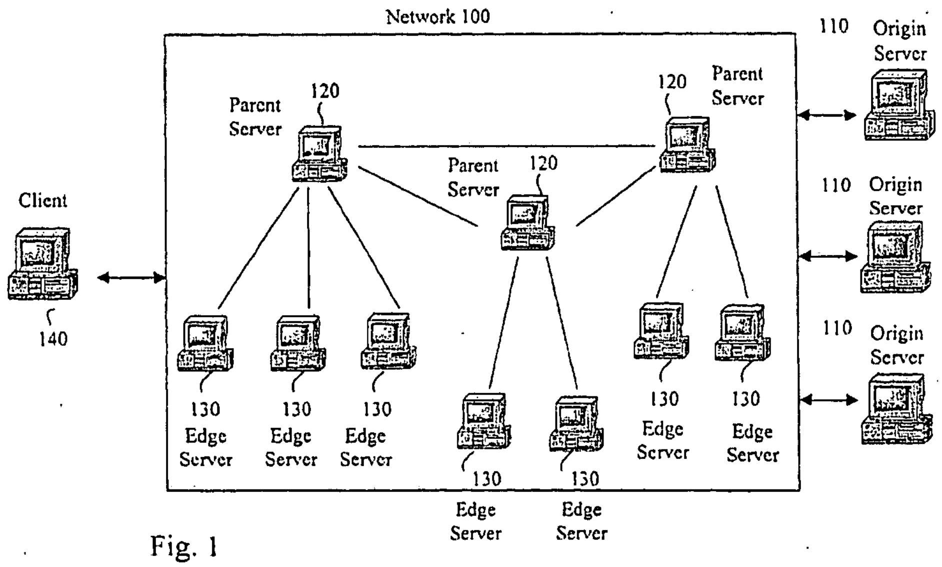


Fig. 1

Note that parent servers 120 are part of the CDN 100 and are distinct from the content sources (origin servers 110). The combination of edge servers 130 and parents servers 120 effectively create an hierarchical CDN.

In one aspect, this invention is a method for managed object replication and delivery. This method is described in the application, e.g., with reference to *data flows* in Fig. 2 (reproduced below) which "depicts embodiments of the method in relation to a portion of the network 100, an origin server 110 and a client 140 . . ." *Specification* ¶0023.

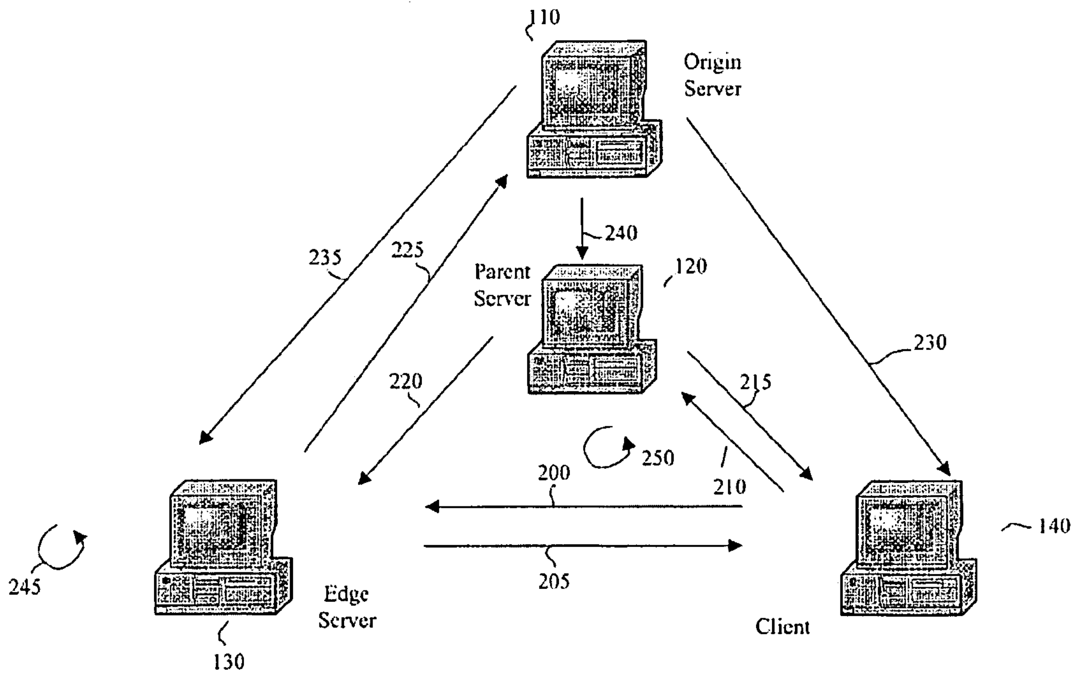


Fig. 2

The method is also described in the application, e.g. with reference to the *flow chart* in Figs. 3(a)-3(b) (also reproduced below).

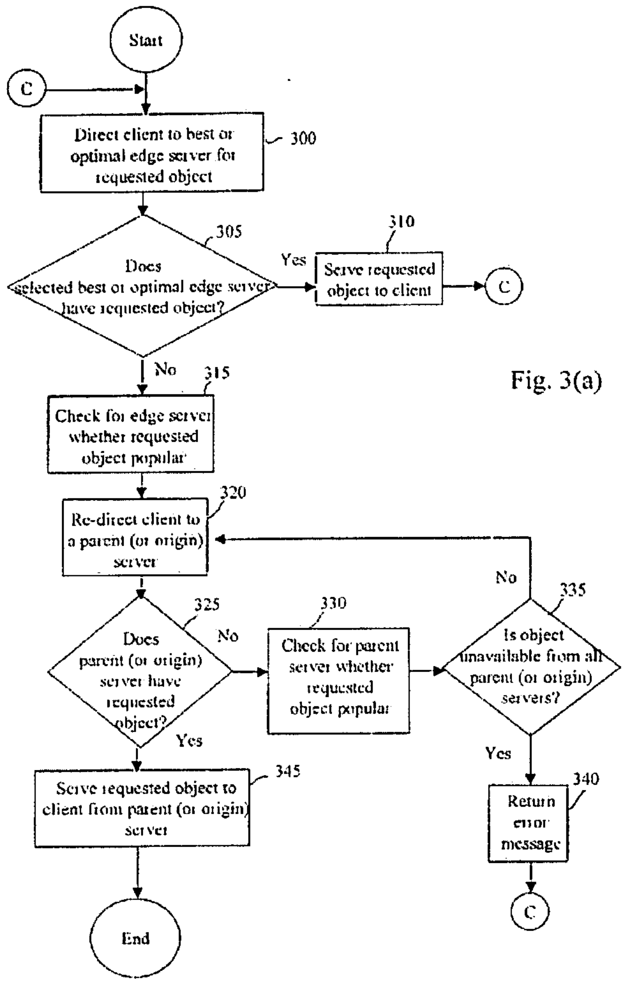

Start
C
Direct client to best or optimal edge server for requested object
300
305
Does selected best or optimal edge server have requested object?
Yes
310
Serve requested object to client
C
No
315
Check for edge server whether requested object popular
320
Re-direct client to a parent (or origin) server
325
Does parent (or origin) server have requested object?
No
330
Check for parent server whether requested object popular
No
335
Is object unavailable from all parent (or origin) servers?
Yes
Yes
345
Serve requested object to client from parent (or origin) server
340
Return error message
End
C

Fig. 3(a)

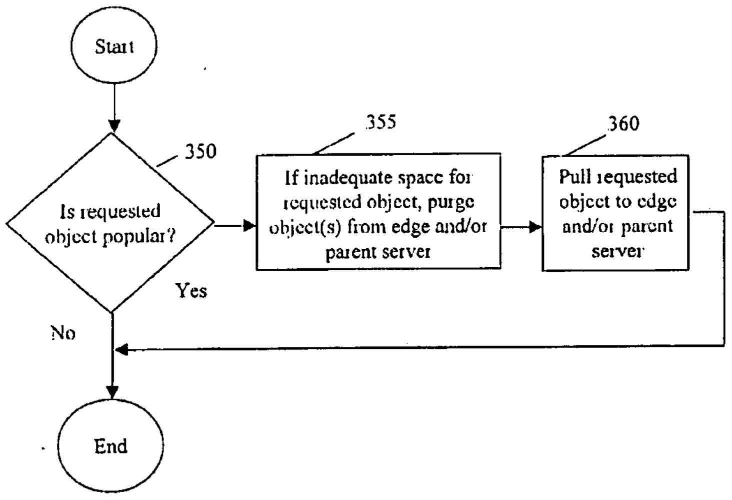

Start
350
Is requested object popular?
355
If inadequate space for requested object, purge object(s) from edge and/or parent server
360
Pull requested object to edge and/or parent server
Yes
No
End

Fig. 3(b)

"Initially, the method . . . directs . . . a client, requesting one or more objects, to an edge server in the network, whether or not the edge server has the requested object(s)." *Specification* ¶0024. "The selected . . . edge server 130 determines . . . whether the edge server already has the requested object and, if so, serves . . . the object to the requesting client 140." *Specification* ¶0025. "If the selected edge server does not have the requested object, a check is initiated . . . for the edge server to determine whether the requested object is popular and if so, to replicate the popular requested object to the edge server." *Specification* ¶0026. "Further, if the selected edge server does not have the requested object, the selected edge server directs . . . the requesting client 140 to a parent server 120. Preferably the client 140 is redirected to a parent server that has the requested object and is able to serve . . . the requested object to the client." *Specification* ¶0028.

So, e.g., as recited in claim 1 and its dependents (claims 2-14), the method includes: Directing a request by a client for an object to an edge server in a network. If the edge server has the requested object, the requested object is served to the client from the edge server. Otherwise (i.e., if the edge server does *not* have the requested object), the client request is redirected to another server, and the requested object is served to the client from that server. If the requested object is popular, the requested object is replicated to the edge server.

Similarly, claims 23-37 recite a computer program product including computer program code to cause a processor to perform the methods for managed object replication and delivery of claims 1-15. Independent claim 16 recites a method similar to independent claim 1, for managed object replication and delivery. Independent claim 16 (and its dependents) recites "directing a request by a client for an object to *an optimal* edge server in a network." Claim 38 recites a computer program product including computer program code to cause a processor to perform the method of claim 16. Independent claim 45 recites a system for managed object replication and delivery including a plurality of edge servers in a network; and a plurality of parent servers in the network. As recited in the claim, at least one of the plurality of edge servers and the plurality of parent servers: direct a request by a client for an object to an edge server in the network, if the edge server has the requested object, serve the requested object to the client from the edge server, otherwise, redirect the client request to another server and serve the requested object to the client from that other server, if the requested object is popular, replicate the requested object to the edge server. Independent claim 59 is similar to claim 45, and recites "at least one of the plurality of edge servers and the plurality of parent servers: direct a request by a client for an object to an optimal edge server in the network."

As noted, if an edge server does not have a requested object (regardless of the current popularity of that object), the client request is directed to another server. ("... the edge servers act as the primary source of serving objects but if a requested object is not available at the edge server a parent server that has the requested object will serve the requested object to the clients" ¶0021). In this manner, the client is not kept waiting *and* non-popular content is not propagated to the edge servers.

B. THE REJECTIONS OF THE CLAIMS SHOULD BE WITHDRAWN

Claims 1-65 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Jungck in view of Sim. Applicant respectfully submits that the obviousness rejections of these claims should be withdrawn because the Examiner failed to establish a *prima facie* case of obviousness for any of the rejected claims.

The Examiner Failed to Make a Prima Facie Case of Obviousness Against Claims 1, 16, 23, 38, 45 and 59

The Examiner states that "[a]s per claims 1, 16, 23, 38, 45 and 59, Jungck disclosed a system . . . comprising: a plurality of edge servers in a network; and a plurality of parent servers in the network (paragraph 19), wherein at least one of the plurality of edge servers and the plurality of parent servers (paragraph 25): direct a request by a client for an object to an edge server in the network (paragraphs. 27 & 35), if the edge server has the requested object, serve the requested object to the client (paragraph 56), otherwise redirect the client request to a server that has the requested object and serve the requested object to the client (paragraph 57)." [*Final Office Action* §3, pg. 2] Applicant respectfully disagrees.

Unlike the presently claimed invention, Jungck performs *on-demand* replication, *regardless of popularity*. In Jungck, if a client request is sent to a server which does not have the requested object, then a copy of that object is *always* replicated on that server. [*See, e.g.,* "The cache server 208 saves/caches Web pages and other content that clients 102, 104, 106, who share the cache server, have requested in the past." *Jungck* at ¶0056.]

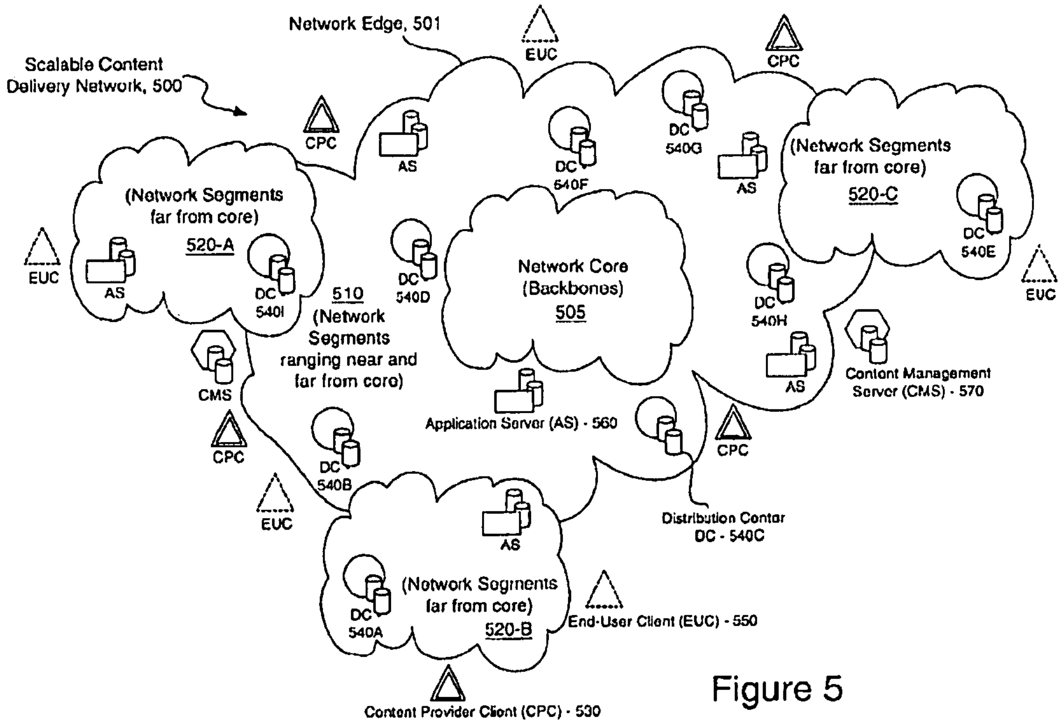
Jungck (in paragraph 0057, relied on by the Examiner) describes the operation of so-called cache servers. (Note that Jungck's "cache servers" are not the same as "edge servers," which Jungck later describes.) "Cache servers 208 invisibly intercept requests for content and attempt to provide the requested content from the cache (also known as a "hit"). ...Where the requested content is not in the cache ..., the cache forwards the request onto the content source. When the source responds to the request ... ***the cache server 208 saves a copy of the content in its cache for later requests***. In the case where a cache server is part of a proxy server, *the cache/proxy server makes the request to the source on behalf of the client* 102, 104, 106. *The source then provides the content to the cache/proxy server which caches the content and also forwards the requested content to the client 102, 104, 106.*", *Jungck* ¶0057, emphasis added.

So, what one of Jungck's cache servers does is this: if it has the requested content then it tries to serve that content. If it does not have the requested content then it forwards the request to the content source or it gets the content from the source and then serves it to the requesting client. In either case, Jungck's cache servers unconditionally replicate the requested content (regardless of its popularity).

The Examiner acknowledged that Jungck does not teach anything about testing for popularity of requested objects or about caching requested objects based on their popularity. [*Final Office Action* §3, pg. 2]

In order to try to overcome this acknowledged deficiency in Jungck, the Examiner has applied Sim. Applicant respectfully submits that Sim does not overcome the acknowledged deficiencies in Jungck.

Sim describes distributing files to a plurality of storage devices in a network. In particular, Sim is concerned with large payload files. Sim describes a system in which a large file may be broken up into parts (partitioned into blocks), and the different component blocks are cached on different ones of the distribution stations. *See, e.g., Sim* ¶0047. Sim describes so-called "distribution stations" at which he keeps all or part of large files. One embodiment of Sim's system is shown in Sim's Fig. 5 (reproduced below).


Network Edge, 501
Scalable Content Delivery Network, 500
EUC
CPC
AS
DC 540F
DC 540G
AS
(Network Segments far from core) 520-C
DC 540E
EUC
(Network Segments far from core) 520-A
EUC
AS
DC 540I
510 (Network Segments ranging near and far from core)
DC 540D
Network Core (Backbones) 505
DC 540H
AS
Content Management Server (CMS) - 570
CMS
CPC
Application Server (AS) - 560
CPC
DC 540B
EUC
AS
Distribution Center DC - 540C
DC 540A
(Network Segments far from core) 520-B
End-User Client (EUC) - 550
Content Provider Client (CPC) - 530
Figure 5

In the embodiment of Fig. 5, content providers upload (publish) data (e.g., files) to the content delivery network 500 using so-called Content Provider Clients (CPCs) 530. Data are uploaded from a CPC 530 to a Content Manager Server (CMS) 570. The CMS processes the data and then provides it to a Distribution Center (DC) 540. That distribution center (DC) then provides some or all of the data to the other distribution centers (DCs) in the CDN. For example, a content provider may upload data via Content Provider Client 530 to Content Management Server (CMS) 570. CMS 570 processes the data and uploads (pushes) it to Data Center (DC) 540H. That Data Center 540H then distributes the data across the other data centers (DC 540A-540G, and 540I) according to various policies. (Sim describes a distribution system in which different DCs get different portions of the data, see, e.g., Fig. 6 and ¶¶0089-90.) All of Sim's distribution is done before there is any end-user involvement. An exemplary distribution center is shown in Sim Fig. 7 (reproduced here):

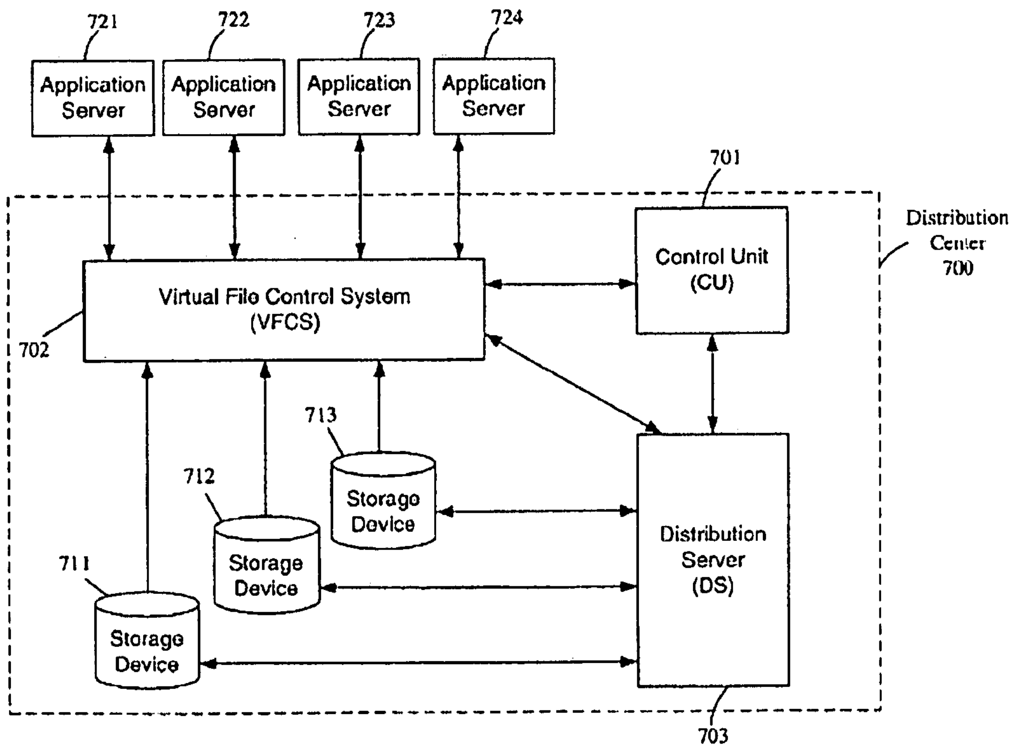


Figure 7

In Sim, end users access data via an End-User Client (EUC) 550 (e.g., a browser) from so-called Application Servers (ASs) 560. "Application servers 721-724 (e.g., streaming servers, FTP servers, and media players), which are ***not part of distribution center*** 700, are shown connected to the virtual file control system 702 ..." *Sim* ¶0088, emphasis provided. When an end-user requests a file from an Application Server, it appears to that user that the Application Server has the entire file ("When an end user connects to application server 721 (e.g., a streaming server), the VFCS creates a virtual appearance that the entire file is available at that node." *Sim* ¶0090.) If there are parts of the file that the application server does not actually have, that application server obtains the needed portions and provides them to the end-user. (*See, e.g., Sim* ¶0171 *et seq.*) Sim does not teach or in any way suggest directing a request from user to another server to get content. In Sim, once a user is getting data from an application server, that application server handles the entire request. The application server may, itself, have to obtain the data from another location, but the user is not directed to that other station. Furthermore, if an application server does get data from another station (in order to serve that data to the user), then that application server unconditionally replicates that data locally (i.e., caches that data) regardless of any popularity of the data.

Another embodiment of Sim's is illustrated in Sim's Fig. 14 (reproduced below). The "Scalable Content Delivery Network SCDN 1400 is essentially the same as SCDN 500 (see FIG. 5) with the individual Distribution Centers and individual Application Servers of SCDN 500 replaced by a plurality of SCDN Stations 1410 and a Central Station 1420." *Sim* ¶0146.

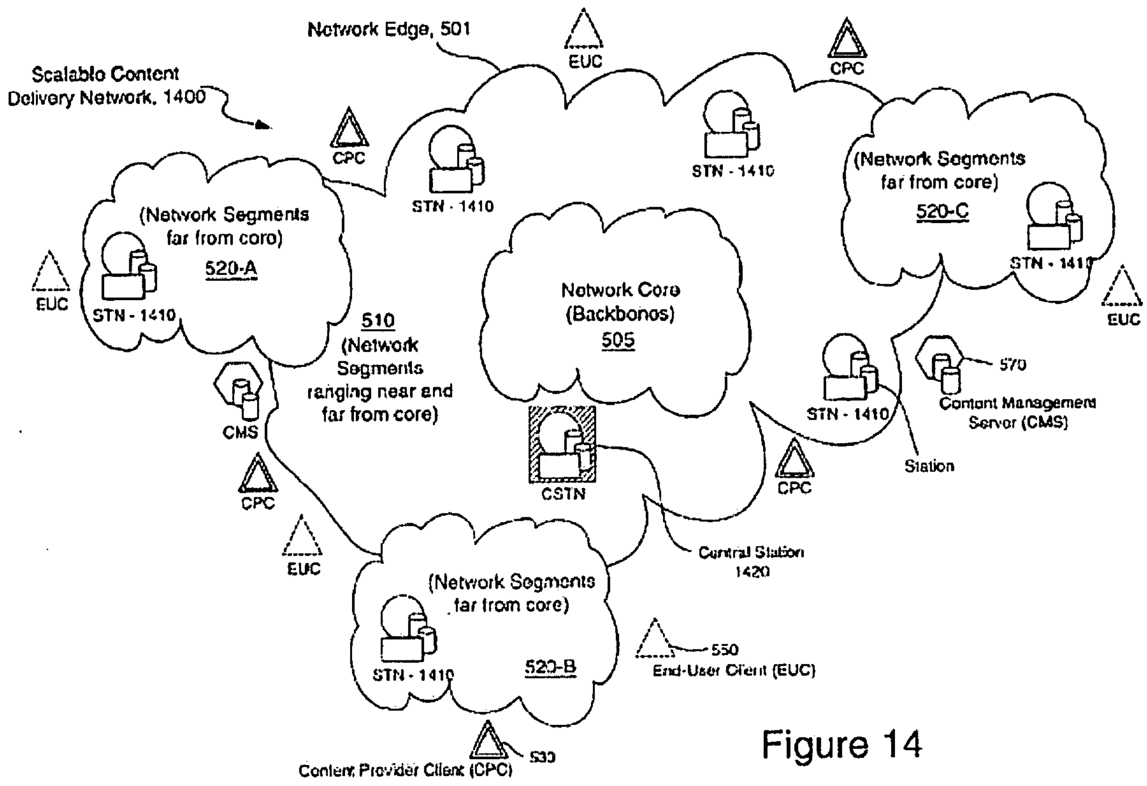

Network Edge, 501
Scalable Content Delivery Network, 1400
EUC
CPC
STN - 1410
(Network Segments far from core) 520-A
(Network Segments far from core) 520-C
(Network Segments far from core) 520-B
510 (Network Segments ranging near and far from core)
Network Core (Backbones) 505
CMS
CSTN
Central Station 1420
570
Content Management Server (CMS)
Station
550
End-User Client (EUC)
530
Content Provider Client (CPC)

Figure 14

In Sim's second embodiment, as with the other embodiment, content provider data files are published (uploaded) to the system in the same manner as before (via Content Provider Clients (CPCs) and Content Management Servers (CMSs). The uploaded data are provided to a Central Station (CSTN 1420) which distributes (pushes) the data to the other stations (STNs). After publication of the data, end-user requests for content are made via End-User Clients (EUCs) 550 (e.g., browsers), via a Station (STN 1410).

Here, again, if a Station 1410 does not have all of the data requested by a user, the Station itself tries to get the data from another source (but the station does not redirect the user to any other source). Note that it is a design goal of Sim that most nodes only contain a portion of a data file. Sim's entire distribution system is directed toward splitting a data file up in various locations (see, e.g., Fig. 13 (reproduced below) and *Sim* ¶0019). Since Sim knows that most nodes only have part of any file (if any at all), it would not make sense to redirect a requesting user to another node (since that other node will likely also only have part of the file).

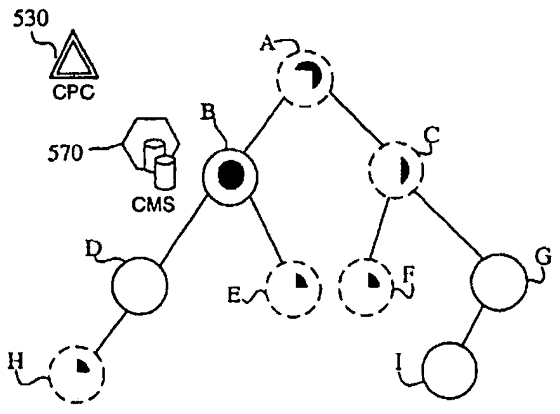


Figure 13

Suppose, e.g., a file is distributed according to the scheme shown in Sim's Fig. 13 (above). If an end-user is directed to node E, that node only has a portion of the file and will have to get the rest of the file from other nodes (e.g., node C and then node B). But if the user were redirected by node E to node C for the second part of the file, then the user would still have to be redirected to node B for the remained of the file. This is not at all what Sim teaches. In Sim, the application server (or station) at node E would itself get the data from the other nodes, keeping a persistent connection with the end-user, while trying to get the needed data. In Sim, an end user may be kept waiting, but will not be redirected.

Sim describes in detail what happens when all the requested data are not present at a station. See, e.g., *Sim* ¶¶00171-176, which states, in part:

> … What a VFCS Server does if all of the requested data is not present in the Station's Storage System is … discussed in the context of FIG. 19.
>
> Assuming that VFCS 1840-3 is processing requests for Application Server 1810-1. If either additional or the full content of the requested file is needed by Application Server 1810-1, VFCS 1840-3 seeks the assistance of a distribution server in the Station's distribution server cluster (e.g., 1510) to retrieve the missing content. …
>
> … The load balancer (e.g., 1720) then selects an

> available distribution server, e.g., DS 1710-2 ..., to service the request. DS 1710-2 issues a search request as a component of Outbound DS Traffic, on behalf of the Application Server 1810-1, to each of its neighbor Stations to locate and download the needed portions of the file. ...
>
> When DS 1710-2 receives reply packets from the neighboring Stations indicating that they contain part of or the entire requested file, distribution servers 1710 in DSC 1510 will download the missing content from those Stations that are least congested and stores it locally in Storage System 1530 ... then processes Application Server 1810-1's request by accessing the data in Storage System 1530 ... and sends data and response back to Application Server 1810-1....
>
> ... During access of a content file via VFCS 1840, if the VFCS detects that all the block files that make up the requested content file are not available locally, it signals the DSC 1510 via an FDP prepare command to download the missing portions from other SCDN nodes. A DS in the DSC 1510 issues an FDP search command to DSs in its neighbor nodes in attempts to locate and download the missing block files .... As the block files are downloaded, the metadata of the content file is updated to register the existence of the block files in the local storage volumes.

Notably, in Sim, as in Jungck, *all* requested content is replicated at the server that is handling a user's request. I.e., in Sim, when a distribution station gets a request for content, that distribution station replicates that content!

Separate and apart from Sim's serving and associated replication process, Sim's provides a so-called Storage Management Agent to administer aspects of the storage system. One function of the Storage Management Agent is to try to maintain sufficient free space on the various storage systems.

> The Storage Management Agent ... determine[s] a reasonable storage safety threshold, adjusts the "popularity" index of a file, and identifies the least likely to be used blocks. A storage safety threshold is the minimum amount of free storage each content provider must reserve at all

times. Based on storage availability and the DS activities, the Storage Management Agent determines the total amount of data to be *pruned* for each content provider and schedules the deletion of the least likely to be used blocks.

Sim ¶0199, emphasis provided.

Sim thus simply uses a "popularity index" as part of a clean-up / management process to try to ***maintain*** sufficient free space on a storage system and so to decide ***what to prune*** in a storage system ("... decides what content to prune ..." *Sim*, Abstract). The "popularity" index mentioned in Sim is not used to decide whether or not to replicate an object that is being served to a user – as noted above, replication is unconditional. As Sim states at ¶0052 (with emphasis added):

> ... the portions and amount of a large payload file ***maintained*** at each node depends on the available storage, popularity of the content, distribution criteria by the content provider, etc. Thus, least likely to be used blocks of a large payload file ***may be pruned*** (i.e., deleted from local storage) *to make room for other highly desirable content.*

See also *Sim* at ¶0230 which states:

> The Storage Management Subsystem watches the available shared storage, the content provider's reserved storage, and the usage logs. It initiates the removal of less popular content to make room for more popular and new content when available storage is running low.

Sim does not, as the Examiner would have it, teach or in any way suggest "*if the requested object is popular*, replicate the requested object to the edge server." Sim, like Jungck, simply and unconditionally replicates the object.

Accordingly, applicant respectfully submits that no proposed combination of Jungck and Sim would produce the presently claimed invention of claims 1, 16, 23, 38, 45, and 59. Any such combination would lack at least the ***conditional***

replication of the present invention, ***based on popularity***. Any such combination would replicate all content, regardless of popularity. To the extent "popularity" is used by any such combination, it would be to determine how to *prune* a cache.

Furthermore, any such combination would lack the claimed "redirecting" in cases where the requested content is not on the server handling the user's request.

All other claims depend (directly or indirectly) from claims 1, 16, 23, 38, 45, and 59, and are therefore patentable for at least the reasons given above.

For at least these reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 2, 16, 24, and 38 are further patentable over Jungck and Sim

In claims 2, 16, 24 and 38, if the edge server does not have the requested object, "the client request [is redirected] to ***a parent server*** in the network that has the requested object and ... the requested object [is served] to the client ***from the parent server***."

As noted above (on page 30), ***parent servers*** 120 are part of the CDN 100 and are distinct from the content sources (origin servers 110). Neither Sim nor Jungck have anything like the claimed "parent servers". In Jungck, if a cache server does not have a requested file, that cache server itself obtains that file from the "content source" (and that file is served to the requestor from that cache server). *Jungck* ¶0057. Jungck's "content source" cannot be considered to be a parent server, as claimed. Jungck's "content source" is not a server in the CDN, it is the content provider's server. In Sim, *all* requested content is served by a distribution station, and if the distribution station does not have the requested content, it gets it from another station (and not from a parent server).

The Examiner cites Jungck ¶0019 to supposedly teach the claimed "parent servers." [*Final Office Action*, §3, pg. 2]. Jungck ¶0019 is reproduced here in its entirety:

> As an introduction, a network interconnects one or more computers so that they may communicate with one another, whether they are in the same room or building (such as a Local Area Network or LAN) or across the country from each other (such as a Wide Area Network or WAN). A network is series of points or nodes 126 interconnected by communications paths 128. Networks can interconnect with other networks and can contain sub-networks. A node 126 is a connection point, either a redistribution point or an end point, for data transmissions generated between the computers which are connected to the network. In general, a node 126 has a programmed or engineered capability to recognize and process or forward transmissions to other nodes 126. The nodes 126 can be computer workstations, servers, bridges or other devices but typically, these nodes 126 are routers.

This paragraph 0019 from Jungck relates to networks generally and has nothing to do with content delivery networks or with parent servers in a CDN.

Since neither Jungck nor Sim has a parent server, the do not and cannot teach or in any way suggest the claimed "redirecting the client request to a parent server in the network that has the requested object and serving the requested object to the client from the parent server."

Further, even if, *arguendo*, one of Jungck's servers (or Sim's servers) is considered a parent server, neither Jungck nor Sim teaches or suggests any sort of redirection of client requests. As noted above, in both Jungck and Sim, all content is served from the same server.

For at least these additional reasons, claims 2, 16, 24 and 38 are further patentable over Jungck and Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 3, 25, 27 and 47 are further patentable over Jungck and Sim

Claim 3 depends from claim 1 and is therefore patentable over Jungck and Sim for at least the reasons given above. Claim 3 recites a method as in claim 1 "wherein redirecting the client request to a server comprises redirecting the client request to a parent server in the network that *does not have the requested object*, ***recursively redirecting*** the request until a parent server in the network having the requested object is reached and serving the requested object to the client from the parent server." Support for this claim can be found in the Specification, e.g., at ¶0029 which states:

> ... if a parent server does not have the requested object, the parent server could itself use a redirection technique recursively (at 325, 335, 320) until a final parent server is reached that has the requested object. The parent server that has the requested object serves (at 215, 345) the object to the client.

As noted above, neither Jungck nor Sim has any notion of the claimed *parent server*. In addition, neither Jungck nor Sim teaches or in any way suggests any kind of redirecting, let alone the claimed "**recursively redirecting"** until a parent server in the network having the requested object is reached and serving the requested object to the client from the parent server." In Jungck the cache server 208 itself goes to the "source" when requested content is not in its cache. *Jungck* ¶0057. There's nothing in Jungck about more than one "source" or about "recursively redirecting" until a source having the object is found. Likewise, Sim lacks any teaching or suggestion of **recursively redirecting** any request until a server having the requested object is found.

Applicant respectfully submits, therefore, that no proposed combination of Jungck with Sim would teach or in any way suggest the claimed: "redirecting the client request to a parent server in the network that does not have the requested object, *recursively redirecting* the request until a parent server in the network having the requested object is reached and serving the requested object to the client from the parent server." Neither Jungck nor Sim has any teaching or suggestion of any kind of recursive redirecting of requests.

Similar arguments apply to claims 25, 27 and 47.

For at least these additional reasons, claims 3, 25, 27 and 47 are further patentable over any proposed combination of Jungck with Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 13, 19, 35, 41, 56 and 61 are further patentable over Jungck and Sim

Further as to claims **13, 19, 35, 41, 56** and **61**, applicant respectfully submits that no proposed combination of Jungck with Sim would teach or in any way suggest the claimed: "***replicating*** the requested object *in accordance with a dynamic replication threshold*." Sim (in ¶0230, relied upon by the Examiner) teaches ***replacing*** cache content based on popularity (less popular content is replaced by more popular content). But Sim is completely silent and lacks any teaching or suggestion of anything like the claimed ***dynamic*** replication threshold. For at least this reason, these claims are further patentable over any proposed combination of Jungck with Sim.

For at least this additional reason, claims **13, 19, 35, 41, 56** and **61** are further patentable over any proposed combination of Jungck with Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 4, 8, 9, 26, 30, 31 48, 52, 53 and 65 are further patentable over Jungck and Sim

In some embodiments, if an object is popular, an attempt is made to replicate it from a *parent server* in the CDN. If the object is not available at a parent server, then the request may be redirected to the *origin server* which serves the requested content to the client. As noted in the Specification at ¶0031:

> ..., where the parent servers collectively are not populated with all of the objects and the network has access to the origin server of a requested object, the client may be redirected (at 225, 320) to the origin server if the requested object is not available on the parent servers. If the origin server has the requested object (at 325), the origin server would serve (at 230, 345) the object directly to the client ...

Claim 4 recites a method as in claim 1, including "... redirecting the client request to an *origin server if* the requested object is *not available at a parent server* in the network and *serving the requested object to the client* ***from the origin server***." As noted above, neither Jungck nor Sim has any notion of the claimed "parent server". In Jungck all replication takes place directly from the content source (the origin server). Jungck does not first look to any other server, let alone to a parent server, before going to the origin server. Sim operates similarly to Jungck.

Similar arguments apply to claim 9 which recites a method as in claim 1 including "...if the requested object is unavailable on parent servers in the network, replicating the requested object to the edge server from an origin server." Neither Jungck nor Sim replicates from an origin server "if the requested object is unavailable on parent servers in the network," as claimed.

Similar arguments apply to claim 8, 9, 26, 30, 31, 48, 52, 53 and 65.

For at least these additional reasons, claims 4, 8, 9, 26, 30, 31, 48, 52, 53 and 65 are further patentable over any proposed combination of Jungck with Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 8, 30, 52 and 65 are further patentable over Jungck and Sim

In some embodiments, when a requested object is popular and is therefore replicated to the edge server, if the requested object is not available on parent servers in the CDN, "the requested object [is replicated] to a parent server in the network from an origin server." See, e.g., claim 8. This is described in the Specification, e.g., at ¶0032, which states, with emphasis provided:

> ... when the edge server determines (at 350) that a requested object is popular but the edge server does not have a copy of the requested object, the edge server initiates the replicating (at 220, 360) of the popular requested object to the edge server from a parent server that has the requested object. Similarly, for example, *when a **parent server** 120 determines (at 350) that a requested object is popular but the parent server does not have a copy of the requested object, the parent server initiates the replicating **(at 240 [in Fig. 2]**, 360) of the popular requested object to the parent server **from an origin server** that has the requested object. ...*

Note, this replication from the origin server to the parent server is in addition to the replication to the edge server.

Neither Jungck nor Sim has any notion of a parent server, and neither Jungck nor Sim, alone or in any proposed combination, teaches or in any way suggests replication to any server other than the server that is handling the request.

For at least these additional reasons, claims 8, 30, 52 and 65 are further patentable over any proposed combination of Jungck with Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 10, 21, 32, 43, 54, and 63 are further patentable over Jungck and Sim

In Sim "a 'popularity' index [is] set by the content provider", and this index is used to control pruning of replicated data. Sim mentions (at ¶0199, see also

¶0197) that the Storage Management Agent "adjusts the 'popularity' index of a file," but Sim provides no indication of when or how this is adjusted, the only indication being that it is set by the content provider "as a prediction of the likelihood of the file to be accessed in the near future." *Sim* ¶0198.

Claim 10 recites a method as in claim 1 "wherein whether the requested object is popular is determined using at least a request rate for the requested object." Sim does not teach or suggesting the popularity of a file being based on a request rate for an object. In fact, *Sim distinguishes popularity from usage* (see, e.g., "Each distribution station is configured to determine how much of the content to save locally, based on information such as ***usage, popularity***, etc." *Sim*, ¶0047, emphasis provided; and "... File Metadata Database holds file metadata ... which includes ... initial *popularity index*, ... actual *usage* rating, ..." *Sim,* ¶0225, emphasis provided).

Similar arguments apply to claims 21, 32, 43, 54, and 63.

For at least these additional reasons, claims 10, 21, 32, 43, 54, and 63 are further patentable over any proposed combination of Jungck with Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 11, 12, 17, 18, 33, 34, 39, 40, 55 and 59 are further patentable over Jungck and Sim

Claim 11 recites a method as in claim 1, including "if an object on the edge server ***is no longer popular***, deleting the object from the edge server." Claim 12 recites a method as in claim 1, including "if an object on the parent server is no longer popular and the object is available on an origin server, deleting the object from the parent server."

Sim teaches pruning data that are not popular, but has no teaching or suggestion of data being "no longer popular" – a dynamic notion of popularity.

Thus, no proposed combination of Jungck and Sim would produce a system which deleted objects that were "*no longer popular*".

Similar arguments apply to claims 12, 17, 18, 33, 34, 39, 40, 55 and 59.

For at least these additional reasons, claims 11, 12, 17, 18, 33, 34, 39, 40, 55 and 59 are further patentable over any proposed combination of Jungck with Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 13, 19, 35, 41, 56, and 61 are further patentable over Jungck and Sim

Claim 13 recites a method as in claim 1 "wherein replicating the requested object comprises replicating the requested object in accordance with a dynamic replication threshold." In both Sim and Jungck replication takes place unconditionally. There is no use in Jungck or Sim of (or even suggestion of) any replication threshold, dynamic or otherwise.

Similar arguments apply to claims 19, 35, 41, 56, and 61.

For at least these additional reasons, claims 13, 19, 35, 41, 56, and 61 are further patentable over any proposed combination of Jungck with Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

Claims 14, 20, 36, 42, 57, and 62 are further patentable over Jungck and Sim

Claim 14 recites a method as in claim 1, replicating the requested object when a popularity of the requested object is greater than a threshold popularity and there is enough storage to replicate the requested object." In both Sim and Jungck replication takes place unconditionally. There is no use in Jungck or Sim of (or even suggestion of) and replication based on any threshold popularity. Nor do either Jungck or Sim teach or in any way suggest the claimed deletion of objects until sufficient space is available.

Similar arguments apply to claims 20, 36, 42, 57, and 62.

For at least these additional reasons, claims 14, 20, 36, 42, 57, and 62 are further patentable over any proposed combination of Jungck with Sim, and for at least these additional reasons the obviousness rejection of those claims under §103 should be withdrawn.

CONCLUSION AND REQUEST FOR PERSONAL INTERVIEW

Applicant respectfully submits that the inventions recited in claims 1-65 are not obvious in view of the cited references and that this application is in condition for allowance. An early action to that effect is earnestly solicited.

The Examiner is kindly requested to contact the undersigned at the number provided to schedule a personal interview to resolve any outstanding issues in this case.

CHARGE STATEMENT: Deposit Account No. 501860, order no. **2711-0040**.

The Commissioner is hereby authorized to charge any fee specifically authorized hereafter, or any missing or insufficient fee(s) filed, or asserted to be filed, or which should have been filed herewith or concerning any paper filed hereafter, and which may be required under Rules 16-18 (missing or insufficiencies only) now or hereafter relative to this application and the resulting Official Document under Rule 20, or credit any overpayment, to our Accounting/ Order Nos. shown above, for which purpose a duplicate copy of this sheet is attached.

This CHARGE STATEMENT does not authorize charge of the issue fee until/unless an issue fee transmittal sheet is filed.

CUSTOMER NUMBER 42624	Respectfully submitted, [illegible] By: /Brian Siritzky/37497 Brian Siritzky, Ph.D., Reg. No. 37497

DAVIDSON BERQUIST JACKSON & GOWDEY LLP
4300 WILSON BLVD., 7TH FLOOR, ARLINGTON, VIRGINIA 22203
MAIN: (703) 894-6400 • FAX: (703) 894-6430